Towards provable security for ubiquitous applications

Mike Burmester, Tri Van Le and Breno de Medeiros

Department of Computer Science, Florida State University
Tallahassee, Florida 323204-4530
{burmester,levan,breno}@cs.fsu.edu

Abstract. The emergence of computing environments where smart devices are embedded pervasively in the physical world has made possible many interesting applications and has triggered several new research areas. Mobile ad hoc networks (MANET), sensor networks and radio frequency identification (RFID) systems are all examples of such pervasive systems. Operating on an open medium and lacking a fixed infrastructure, these systems suffer from critical security vulnerabilities for which few satisfactory current solutions exist, particularly with respect to availability and denial-of-service. In addition, most of the extant knowledge in network security and cryptography cannot be readily transferred to the newer settings which involve weaker devices and less structured networks.

In this paper we discuss the security of pervasive systems and focus on availability issues in malicious environments. We articulate a formal security framework that is tuned for the analysis of protocols for constrained systems and show how this can be used with applications that involve MANET and RFID systems. In our approach we shall use optimistic protocols for which the overhead is minimal when the adversary is passive. When the adversary is active, depending on the application, the additional cost is either used to trace malicious behavior or born by non-constrained components of the system. We consider mechanisms that will support self-healing and promote a fault-free system state, or a stable system state, in the presence of a Byzantine adversary.

Keywords: Ubiquitous applications, RFID, MANET, fault tolerance, tracing malicious faults.

1 Introduction and motivation

We give an overview of the security of pervasive systems, focusing on availability issues in the presence of Byzantine faults. Our goal is to specify formal simulation frameworks for analyzing security objectives and to describe our designs of mechanisms and algorithms that achieve proven availability, uninterrupted services, high efficiency and low overhead in such systems. With a large amount of research already invested into other non-security issues, it is preferable whenever possible to design mechanisms that integrate security into existing algorithms that are well established in the literature.

The emergence of computing environments where smart devices are embedded pervasively in the physical world has made possible many interesting applications and triggered several new research areas. These include pervasive systems with constrained resources that intelligently configure and connect themselves, in particular, mobile ad hoc networks (MANETs), sensor networks and RFID systems. Nevertheless, the deployment of such systems in practice poses great challenges concerning their security and robustness in the presence of malicious faults. So far, research has focused on functionality, performance and services, with security being given a lower priority and centered mainly on confidentiality and integrity, but not availability (under malicious attacks). In particular, most research on network security and cryptography involves highly powered, highly structured redundant systems. For this reason the proposed security solutions are often inappropriate for these networks. This is particularly true regarding Denial-of-Service (DoS) attacks against power constrained systems. Furthermore, most practical security patches are only for specific attacks, leaving an unjustified belief that proven security and efficiency are conflicting goals.

Paper outline. We discuss secure mechanisms and protocols for ubiquitous systems. We consider two applications.

1. Secure Mobile Ad hoc Networks (MANET). In particular we consider:
   - A formal simulation framework for security that focuses on availability under DoS attacks, and that allows for concurrency and universal composability.
   - Mechanisms that provably support availability in malicious environments within the formal simulation security framework.
   - Protocols that provide message-delivery guarantees and that provably support network self-healing from malicious attacks. In particular, protocols that enable migration of the network to a fault-free state.
2. Secure Radio Frequency Identification (RFID). In particular we discuss:
   - A formal simulation framework for security of RFID systems that models availability, privacy, and authenticity services, and
   - Provably secure scalable anonymous authentication protocols for RFIDs in the formal framework.

Our approach is holistic and promotes self-healing. It provides for novel optimistic mechanisms that deal with security and availability issues in an efficient manner with low overhead. We focus on systems that will tolerate, trace and eliminate faults by reconfiguring. The threat model allows for a very powerful Byzantine adversary: malicious nodes are not bounded by the system specifications and can use covert channels, more powerful transmitters/receivers, responders etc. Security will be proven using a well established cryptographic framework. In contrast to traditional cryptographic solutions, our approach is suitable for low power, low cost devices in a malicious environments with no infrastructure.

A self-healing strategy for ubiquitous systems. Self-healing is achieved by tracing malicious behavior. Whenever a component exhibits non-system faulty behavior, a tracing mechanism is activated and the component isolated. Assuming the number of potential non-system faults is bounded (the Byzantine assumption), the system ultimately will be fault-free. By using low level (optimistic) mechanisms, there is no extra cost when the adversary is passive. Even when faults do occur, the overhead involved is small. This makes it possible to achieve our security goal with practical systems that have constrained resources.

2 Securing MANET applications

2.1 A Formal simulation framework for security

There are several ways to capture the unpredictable nature of a mobile ad hoc network. Whichever way is used, there are important mobility and medium aspects that must be reflected. In its simplest form, a *mobile ad hoc network* is a stochastic process $\mathcal{G} = \mathcal{G}_1, \mathcal{G}_2, \dots,$ where $\mathcal{G}_t$ is a random graph with node set V, for which communication is: (i) *synchronous*, the time for a single transmission to be received is bounded by a constant; (ii) *promiscuous*, a packet transmitted by node will be received by all its neighbors. Links can be undirected (the neighbor relationship is symmetric) or directed (the neighbor relationship is asymmetric). We note that to the best of our knowledge, to date, all routing algorithms proposed in the literature support only bidirectional or undirected links.

The stochastic aspect of $\mathcal{G}$ is determined by the states of the nodes of $\mathcal{G}$ and Nature. Nature's contribution comes from the environment and the fact that the communication is wireless. A wide variety of factors may affect the communication, ranging from weather to radio interference and physical obstacles.

Adversary structure. In our model for the adversary, malicious nodes are not bounded by the constraints of the mobile ad hoc network. In particular, the adversary can send packets to arbitrary nodes and eavesdrop on all communication (not necessarily via nodes under his control). In this respect, our model differs from other models.

Definition 1. Let $\mathit{\Gamma}$ be a family of subsets V' of the node set V. We call $\mathit{\Gamma}$ an *Adversary Structure* [21]. The adversary $\mathcal{A} = \mathcal{A}_{\mathit{\Gamma}}$ selects a subset $V' \in \mathit{\Gamma}$ and can corrupt all its nodes during the lifetime of the system.[1] $\mathcal{A}dv$ controls the nodes of V' and may use them to undermine the security of the network. We call these nodes *corrupted* or *faulty* and refer to $\mathcal{A}dv$ as a $\mathit{\Gamma}$-adversary. The adversary may be *passive* or *active*. A *passive* adversary (also called *honest-but-curious*) will only eavesdrop on the network communication. An *active* adversary may use the corrupted nodes to prevent the normal functioning of the network via *snooping, dropping, modifying*, and/or *fabricating* network messages. Nodes that are actively involved in such attacks and the corresponding faults are called *malicious* or *Byzantine*. Malicious nodes may use hidden (covert) channels or "wormholes" through which they can communicate or tunnel packets.

A particular case of the Adversary Structure model is the *Byzantine faults* model [37] for which $\mathit{\Gamma} = \{V' \subset V \mid |V'| \leq k\}$, for some threshold k. In this case the adversary $\mathcal{A} = \mathcal{A}_k$ can control up to k nodes. We call $\mathcal{A}_k$ a k-adversary.

Simulation framework. Our security simulation framework follows cryptographic paradigm for the security of network protocols [3]. Mobile nodes are probabilistic Turing machines with a special transceiver tape linked to a network oracle $\mathcal{O}_{\mathcal{G}}$. For concurrent executions of distributed algorithms, we use a model with an infinite collection of oracles that emulate concurrent sessions of the algorithm, with which the adversary can interact. The following definition captures the basic security requirements of this approach for secure distributed applications. For a more formal discussion of a simulation framework for security, see Section 5.

Definition 2. Let $\mathcal{G}$ be a mobile ad hoc network and π a distributed algorithm of $\mathcal{G}$.

- $\mathit{\Gamma}$-*availability* holds for π if $Prob[\pi$ fails in presence of $\mathcal{A}]$ is negligible for all $\mathit{\Gamma}$ adversaries $\mathcal{A}$.
- $\mathit{\Gamma}$-*tolerance* holds for π if $|Prob[\pi$ fails in presence of $\mathcal{A}] - Prob[\pi$ fails in the absence of $\mathcal{A}]|$ is negligible for all $\mathit{\Gamma}$ adversaries $\mathcal{A}$.

The probabilities are taken over the adversary's and honest parties' coin tosses.

[1] There are several generalizations of this model. One such generalization allows $\mathit{\Gamma}$ to be dynamic: at regular intervals $\mathcal{A}dv$ can replace V' by $V'' \in \mathit{\Gamma}$, that is, release the nodes of $V' \backslash V''$ and replace them by the nodes of $V'' \backslash V'$. Another generalization involves hybrid faults: malicious faults and physical faults. We shall not consider these models here.

2.2 Security issues for routing algorithms

Communication in an ad hoc network is achieved by forwarding packets via paths. Depending on where most of the routing effort takes place, there are two types of routing: *network-centric* and *source-centric*. With network-centric routing (such as DSDV [32], WR [6] and AODV [33]) the routing effort is distributed within the network; with source-centric routing (such as DSR [27]) most of the routing effort is done by the source node.

From a security point of view, network-centric routing requires substantial cooperation between network nodes and strong trust relationships. These algorithms are therefore more vulnerable to malicious faults. On the other hand, with source-centric routing, the source is less dependent on intermediate node cooperation and thus less vulnerable to malicious attacks.

Denial-of-Service attacks. A DoS can be triggered in several ways, as for example by: (1) Flooding in dense networks; (2) Flooding irrelevant packets; (3) Packet dropping, *e.g.*, on routers; (4) Preventing route discovery.

Man-in-the-Middle attacks. In a MiM attack the adversary takes control of the communication channel between the source and destination by interposing between them. Active MiM attacks include, for instance: (1) Wormhole attacks [26]; (2) Rushing attacks [26]; and (3) Sybil attacks [18].

Security at the physical and data link layers. There are two types of faults that may occur in a routing algorithm: faults whose effect is stochastically indistinguishable from ordinary link failures caused by the mobility of the system, radio interference, power failure etc, and faults whose effect can be distinguished. Faults that do not deviate from ordinary failures are best dealt with at the physical or data link layer of the protocol stack with Medium Access Control protocols. At these layers one can also deal with jamming attacks (using frequency-hopping spread spectrum techniques) and most isolated DoS attacks.

Faults of the second type, although by definition statistically detectable, can be quite hard to trace or locate. They include malicious faults, which may occur when they are least expected, and may not be traceable with statistical failure analysis. The reason for this is that any analysis based on reported failures can be manipulated by the adversary. Faults of this type have to be addressed at higher levels.

Security issues of Ariadne, SEAD and SAODV. Several routing protocols in the literature address security issues (see *e.g.*, [31]). Examples that are well established in the literature include: Ariadne [24], SEAD [25] and Secure AODV [39]. These algorithms do not tolerate insider faults caused by packet dropping or by colluding nodes (on paths). In particular they do not tolerate wormhole attacks.

Tolerating DoS Attacks with cell grids and colored graphs. A cell-grid approach –see Figure 1, is proposed in [9] in which only one node in each cell is only needed to propagate a packet. This approach guarantees packet delivery during DoS attacks including malicious DoS attacks. This result is based on circular broadcast ranges. It is possible to remove this restriction and extend the cell-grid approach to routing protocols (for example, by taking a succession of adjacent cells as a virtual path).

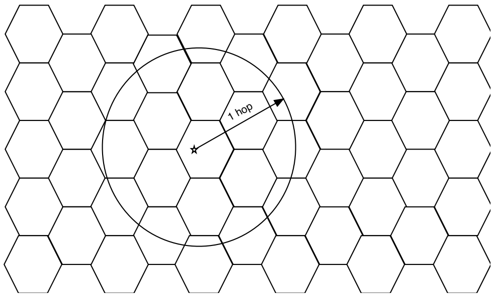

Fig. 1. The cell-grid and a node with its broadcast range.

2.3 An optimistic algorithm that traces malicious faults

Here we describe an optimistic algorithm that will trace malicious faults [7]. For this algorithm there is no additional cost when there are no faults. When faults do occur, the cost to locate a fault is one tracing round and two digital signatures (for a short *probe* and a short $failreport$). In either case, a packet is confirmed successfully delivered, or a fault location is determined with only two digital signatures. This is the most efficient routing algorithm that will trace malicious behavior even when faulty nodes collude. It improves on the tracing algorithm in [2] that requires at least $\log(n)$ communication rounds and signatures to locate a malicious fault, and that does not consider collusions. In our algorithm faults that can be dealt with at the data link layer by error correction and re-sending packets are treated as non-malicious. The protocol is described in Figure 2 and Figure 3. The following notation is used:

- $pkt_{sd} = [[s, d, sn, seq_s, data]]_{sd}$: a packet consisting of identifiers s, d, a session number sn for tracing algorithm (unique to each session), the sequence number seq_s for pkt_s, and $data$, authenticated with the key shared by s, d.
- $ack_{sd} = [[s, d, sn, seq_s]]_{sd}$: an authenticated acknowledgment.
- $prob_s = [s, d, sn, seq_s, hash(pkt_{sd}), hash(ack_{sd})]_s$: a digitally signed probing request by s.
- $failreport_y = [s, d, y, succ(y), sn, seq_s]_y$: a digitally signed failure report by y.
- $timer_{xy}$: a bound on time taken for a round trip from x to y for pkt_s.
- $prec(x)$, $succ(x)$: the node that precedes, succeeds x on the path taken by pkt_s.

In the protocol, the source s sends a packet pkt_{sd} to $succ(s)$ to be delivered to the destination d. If there are no faults then the packet reaches d and s will receive an authenticated acknowledgment ack_{sd}. If there is a fault the source s will send $prob_s$ with details of pkt_{sd} and ack_{sd} requesting from intermediate nodes to compare these values with their stored values. If a fault is detected by an intermediate node x then a $failreport_x$ is issued and send upstream to s. Each intermediate node x node sets timers $timer_{xd}$ and $timer_{xs}$, to determine if and when a $failreport_x$ should be issued. Thus if $succ(x)$ is not faulty, x will receive from $succ(x)$ after $x \xrightarrow{pkt_s} succ(x)$ and $x \xrightarrow{prob_s} succ(x)$ and before $timer_{xd}$ timeouts a valid $failreport_y$.

Observe that when there are no faults s and d only check the validity of pkt_{sd} and ack_{sd}, and intermediate nodes only forward pkt_{sd} and check the header of ack_{sd}.

Theorem 1 ([7]). *For any Γ-adversary, the tracing algorithm in Figures 2 and 3 either delivers pkt_s to the destination d or will trace at least one faulty node.*

```
$Source\ s.$ Set $seq_s = 0$. While a connection to $d$ has not terminated do:
 1. Set $timer_{sd}$ and send $pkt_{sd}$ to $succ(s)$.
 2. If a valid $ack_{sd}$ for $pkt_{sd}$ is received before timeout then set $seq_s = seq_s + 1$.
 3. Else set $timer_{sd}$ and send $prob_s$ to $succ(s)$.
    (a) If a valid $failreport_y$ for $pkt_{sd}$ is received before timeout then $y$ or $succ(y)$ is malicious;
    (b) Else $succ(s)$ is malicious.

$Destination\ d.$ When a valid $pkt_s$ is received:
 1. Construct and send $ack_d$ to $prec(d)$.
```

Fig. 2. An optimistic tracing algorithm, I.

```
$Intermediate\ node\ x.$ When $pkt'_{sd}$ is received:
 1. Set $timer_{xs}$, and send $pkt'_{sd}$ to $succ(x)$.
 2. If an $ack'_{sd}$ is received then send it to $prec(s)$.
    (a) If a valid $probe_s$ for $pkt'_{sd}$ is received with $ack_{sd} \neq ack'_{sd}$ before $timer_{xs}$ times out,
        set $timer_{xd}$ and send $probe_s$ to $succ(x)$.
         i. If a valid $failreport_y$ for $pkt'_{sd}$ is received before $timer_{xd}$ times out:
            send $failreport_y$ to $prec(x)$;
        ii. Else construct and send $failreport_x$ to $prec(x)$.
 3. Else if a valid $probe_s$ for $pkt'_{sd}$ is received with $pkt_{sd} = pkt'_{sd}$ before $timer_{xs}$ times out,
    reset $timer_{xd}$ and send $probe_s$ to $succ(x)$.
    (a) If a valid $failreport_y$ for $pkt'_{sd}$ is received before $timer_{xd}$ timeouts:
        send $failreport_y$ to $prec(x)$;
    (b) Else construct and send $failreport_x$ to $prec(x)$.
```

Fig. 3. An optimistic tracing algorithm, II

Tracing malicious behavior with AODV and DSR. Most of the routing algorithms can easily be extended to incorporate our tracing mechanism in the communication phase. For example, for distance vector based routings such as DSDV, AODV, and DSR , malicious faults will be traced by using the optimistic tracing algorithm for packet processing (the store-and-forward process). This can be done at the network layer, *i.e.*, after error checking at the data link layer (MAC).

2.4 Adaptive Multipath Routing

In this section we consider secure distributed algorithms for finding maximal vertex disjoint paths that we have used to design secure AODV-type algorithms [33].

Multipath routing involves the establishment of multiple paths between source and destination pairs. These paths may be used for redundant communication to control malicious attacks. A major advantage in using multipaths is that, by exploiting redundancy we can guarantee service continuity, even when the adversary is active.

An Adaptive Multipath Routing algorithm Finding routes with multiple paths in networks that do not have a fixed infrastructure is a challenge and in general requires a different approach to that used with fixed infrastructures. An adaptive multipath routing algorithm that combines in parallel a distributed version of Ford-Fulkerson Max-Flow algorithm [19] (at the source) with a local network discovery algorithm (for nearby nodes) to find vertex-disjoint paths that link the source to the destination is proposed in [7]. This algorithm is proven secure in [7]. It is shown that:

Theorem 2. *The adaptive multipath routing algorithm in* [7] *tolerates any k-adversary, provided that the network graph is $(k+1)$-connected, $k \geq 1$.*

The novelty of this algorithm is that it is resistant to malicious DoS attacks which are addressed adaptively. In particular, when there are no attacks a single route is used. With each malicious attack, the multipath is adaptively reconstructed to deal with the threat. Only the shortest route(s) is (are) actually used, while the rest are kept alive. Furthermore, communication is activated as soon as a path becomes available, so there are no unnecessary delays.

In general when faults in a t-multipath occur beyond a certain acceptable threshold, the source s will use a $(t+1)$-multipath. Since the new set of paths is already constructed in the background, the delay caused by faults is minimized. Most of the time, there should be no delay. Furthermore, in our algorithm, the set of vertex-disjoint paths of the multipath is constructed incrementally, so that even when delays are unavoidable, they are minimal.

Observe that having local information available centrally is easier than having it distributed. In particular, the procedure used in the adaptive routing algorithm by the source allows more vertex-disjoint paths to be found than in most other multipath routing protocols. As a consequence fewer communication rounds may be needed when faults occur.

This algorithm can be can combined with the Dynamic Source Routing algorithm [27] to get an adaptive multipath DSR algorithm for reliability and service continuity in the presence of malicious adversary. Similarly, we may combine the adaptive multipath routing algorithm with the tracing mechanism to get an adaptive routing algorithm that will trace malicious behavior.

3 Securing RFID Applications

Radio Frequency Identification Devices (RFIDs) were initially developed as very small electronic hardware components having as their main function to broadcast a unique identifying number upon request. The simplest types of RFIDs are *passive tags*, that do not contain a power source, and are incapable of autonomous activity. These devices are powered by the reader's radiowaves, and the antenna doubles as a source of inductive power. Active tags, on the other hand, contain a power source and transmitter, and are capable of autonomous communication. Examples of such tags are toll passes. The low cost and high convenience value of RFIDs give them a potential for massive deployment, and it is expected that they will soon outnumber all other computing device types. Consequently, RFIDs are increasingly used in applications that interface with information security functions.

RFIDs are a challenging platform from an information assurance standpoint. Their extremely limited computational capabilities imply that traditional techniques for securing communication protocols cannot be used with such devices, and instead that new, lightweight approaches must be considered. Yet the privacy and security requirements of RFID applications can be quite significant. Herein we describe measures for the provision of security and privacy that are feasible for RFID applications. Ultimately, this should be accomplished with as rigorous a view of security as other types of applications.

We discuss protocols for RFID applications that:

1. are provably secure under formal simulation frameworks that capture the behavior of honest and adversarial parties, and that articulate a comprehensive security view in terms of an ideal functionality.
2. explicitly consider the existence of side-channels, an issue often ignored in modelling security of traditional applications, but which can be critically important in the RFID context; and
3. are computationally lightweight, taking into consideration the hardware-imposed constraints of the medium.

3.1 History of a Provably Secure RFID Authentication Protocol

In order to illustrate the need for comprehensive security models for RFID applications, we consider variants of the HB protocol which have been proposed as a practical form of secure RFID authentication. Introduced in [23], the HB protocol was originally designed for use by humans—who, as RFID tags, have limited ability to perform complex computations. In RFID tags, the HB protocol leverages the fact that generation of reasonably strong random numbers can be done cheaply by exploring physical properties—ultimately, exploiting the principle of little separation between hardware and software in the RFID domain.

The HB protocol can be proven secure against passive adversaries in a non-concurrent protocol execution setting by a simple reduction to the so-called "Learning Parity with Noise" (LPN) problem. However, it is completely insecure against an active adversary. To fix these problems, the HB protocol was adapted to include challenges from both readers and tags, leading to the HB+ protocol. Protocol HB+ can be proven secure against active adversaries [28] in a simplified model where the adversary is a malicious reader attacking a single honest tag. The proof has been generalized to a parallel and concurrent setting in [29] showing that rewinding techniques in the original security proof in [28] are not truly necessary; once re-winding is eliminated, one may claim as in [29] that multiple simulations can be executed simultaneously without exponential amplification of the probability of simulation failure, and therefore that the scheme is secure in the concurrent setting.

Both of the above security results are established in a simple attack model, which is not a multi-party model. They cannot be held as providing evidence that the scheme is secure in practical applications, where the adversary may communicate with both readers and tags simultaneously. Indeed, man-in-the-middle attacks do exist [20] and result in a total protocol break, as we now describe. In what follows, we denote bit vectors in boldface, and if $\mathbf{v}$ is a bit vector, by $|\mathbf{v}|$ we mean the Hamming weight (number of non-zero components) of $\mathbf{v}$. If $\mathbf{a}$ and $\mathbf{b}$ are two bit vectors of length k, denote by $\mathbf{a} \cdot \mathbf{b}$ the value $a_1 \wedge b_1 \oplus \cdots \oplus a_k \wedge b_k$, where $\oplus$ denotes the XOR bit operation. Legitimate readers and tags share a pair of keys $(\mathbf{x}, \mathbf{y})$.

The attack works as follows: The attacker first chooses a k-bit string $\mathbf{d}$ at random and XORs it against the Reader's challenge $\mathbf{a}$ before forwarding it to the tag. Since the same value $\mathbf{d}$ is used in all protocol rounds until an outcome is produced by the Reader, it is easy to show that the authentication will be successful with overwhelming probability when $\mathbf{d} \cdot \mathbf{x} = 0$—for in that case the responses by the Tag are the same as in the regular protocol. The opposite will be true when $\mathbf{d} \cdot \mathbf{x} = 1$, i.e., the Reader will reject with overwhelming probability. The adversary has then learned one linear relation among the bits of $\mathbf{x}$, and by repeating the attack with different values of $\mathbf{d}$, the full value of $\mathbf{x}$ can be recovered. The adversary can also learn the key value $\mathbf{y}$ by performing a similar attack, where the value $\mathbf{d}$ is xored into the blinding factor $\mathbf{b}$ instead of into the challenge $\mathbf{a}$.

The attack illustrates the fact that security proofs, when carried out in an overly simplified attack model, fail to convey implications for the practical security of protocols. The above example-cum-moral-lesson-tale is a compelling one from the realm of recent RFID protocols, but the basic premise that protocol analysis should ideally be carried out within a comprehensive attack model has been well recognized—e.g., see [12] for general arguments in this regard.

There exist competing approaches for analyzing the security of protocols against arbitrary adversarial configurations. Some of these have been successful at other areas of computer science and adapted for the purposes of security analysis, such as techniques based on model-checking, and formal methods. We will take the approach of indistinguishability between real and ideal protocol simulations. This formalism is based on the premise that one should first define an *ideal functionality* for the protocol—i.e., how to achieve security in an ideal world where the honest parties have a secure communication channel to a trusted party. Then, one constructs a reduction that maps real protocol runs to protocol runs in the ideal world, and shows that honest parties cannot distinguish real and ideal protocol executions. The above formulation was introduced by Beaver [5, 3, 4], and extended by Canetti as the universal composability framework [11–13]. It has also been ported to a modular, formal models-type approach called *reactive systems*, which emphasizes independent analysis of cryptography and communication layers, by Pfitzmann and Waidner [34, 35].

Formal modelling of protocols and cryptographic primitives via real vs. ideal simulations is an increasingly respected paradigm for the analysis of multi-party protocols, including authentication and key-exchange [16, 22, 15], zero-knowledge proofs [14, 17], and the universe of cryptographic primitives [30]. More recently, an RFID privacy-oriented protocol has been proven secure in a strong real/ideal setting [1]. In Section 5, we introduce a first attempt at a comprehensive security model for RFID applications. We demonstrate that the man-in-the-middle attack to the HB+ protocol is captured in our model and therefore, that HB+ cannot be proven secure in that model. We also argue that the model is a reasonable framework to study the real-world security of RFID applications.

4 Towards practical secure anonymous RFID authentication

An RFID authentication system has three components: tags T, readers R, and a trusted server $\mathcal{S}$. Tags are wireless transponders: they have no power of their own and respond only when they are in an electromagnetic field. Readers are transceivers and generate such fields, which they use to transmit challenges to tags (via wireless broadcast). There are two types of challenges: multicast and unicast. Multicast challenges are addressed to all tags in the range of a reader, whereas unicast challenges are addressed to specific tags. In our protocols below we consider both types of challenges. However, our multicast challenges are just random strings, and *all* tags in the range of a reader R are challenged with the same random string. This kind of action is *not* usually counted as a communication pass in an authentication protocol.

We shall assume that all honest tags T adhere to the system specifications and the requirements of the authentication protocol. The same applies to the honest readers R, and to the trusted server $\mathcal{S}$. Tags are issued with individual private keys K which they share only with the trusted server $\mathcal{S}$. These keys are used by the tags for authentication. We denote by $\mathcal{K}$ the set of all authorized keys (issued by $\mathcal{S}$).

In our RFID authentication protocols we shall assume that honest readers R and the server $\mathcal{S}$ are linked by a secure communication channel (reliable and authenticated).

4.1 A Provably Secure 1-Pass Optimistic Anonymous RFID Authentication Protocol

In Figure 4 we describe a one-pass protocol [10] that authenticates RFID tags anonymously, and that is *optimistic*, in the sense that when the adversary is not active the cost to both the tag and the server is minimized. This protocol—named O-TRAP for optimistic, trivial RFID authentication protocol—was introduced in [10], and motivated by security issues identified by G. Tsudik in his proposed YA-TRAP protocol [38]. In [8], a security framework and full proof of security for O-TRAP is given.

R broadcasts r_{sys}

1. $T \rightarrow R \rightarrow \mathcal{S} : r_{tag}, \;\; h = H_K(r_{sys}, r_{tag})$

T updates $r_{tag} = H_K(r_{tag})$

$\mathcal{S}$ accepts T if:

- $\exists \; (r_{tag}, K) \in L$ s.t. $h = H_K(r_{sys}, r_{tag})$, or
- $\exists \; K \in \mathcal{K}$ s.t. $h = H_K(r_{sys}, r_{tag})$.

$\mathcal{S}$ updates $r_K = H_K(r_{tag})$ in the look-up table L.

Fig. 4. A one-pass anonymous RFID authentication protocol. Figure 5 shows the lookup table L.

In O-TRAP, each reader R broadcasts a random string r_{sys} obtained from the server $\mathcal{S}$, and updated at regular intervals. All the tags in the range of R use the same r_{sys}, but will combine it with a locally generated string r_{tag}, and send (broadcast) to the reader R the MAC: $h = H_K(r_{sys}, r_{tag})$. Here $H_K(\cdot)$ is a keyed hash $H(K, \cdot)$. T computes the value of local string r_{tag} by taking the MAC of its previous value, stored locally. The server also updates the value r_K in a local key look-table –see Figure 5. From this table, and the value r_{tag} sent by T, the server can find a corresponding key K' and check that the value h is that same as $H_{K'}(r_{sys}, r_{tag})$. If the tag T has not been challenged by an unauthorised reader, the value h will be correct. In this case the cost for both the tag and the server is two MACs. However, if the tag has recently interacted with a malicious reader, the stored values will be out-of-sync. Then the server will have to exhaustively search through all keys $k \in \mathcal{K}$ to find the correct value and resynchronize. Note that in the dishonest case the extra computational cost is borne out by the server and not by the tag. By exploiting the higher computational capabilities of the trusted server, we have designed a strong authentication protocol that provably hides the identity of tags from eavesdroppers and malicious readers without requiring the tag to ever perform expensive public-key operations. In

strings	r_{K_1}	r_{K_2}	$\cdots$	r_{K_n}
keys	K_1	K_2	$\cdots$	K_n

Fig. 5. The key look-up table L.

all cases, the tag only needs to compute two MACs to authenticate itself. In the honest case, this is also the protocol cost for the central server.

Theorem 3 ([8]). *The 1-pass optimistic anonymous protocol is available, anonymous and secure in the security framework defined below (Section 5).*

To the best of our knowledge, O-TRAP is the only anonymous, strong RFID authentication protocol that is also amenable to being proven secure within a comprehensive adversarial model, which we describe in the next section.

In the following section we formalize the security definitions for RFID protocols. The model largely follows existing paradigms for security of general-purpose network protocols, but becomes specific to the context of RFID applications in two aspects. First, we consider *availability* explicitly, capturing security against unauthorized disabling of tags directly within the model.

Secondly, we restrict concurrency by prohibiting tags from executing more than one session at a time. Note that this is a restriction only on individual, honest tags—many honest tags can be executing concurrently. In addition, readers (whether honest or corrupt), the central server, and dishonest tags can execute multiple sessions simultaneously. Yet, the requirement that a single honest tag can participate only in one session at a time facilitates the design of concurrently secure protocols. As the restriction is a mild one, and in accordance with the capabilities of RFID technology, it is beneficial in that it enables designers of security protocols to concentrate on the crucial security aspects and on how to balance competing interests, such as requirements of low computational cost and low memory utilization.

Proof structure. The proof (omitted here) consists of two stages. First, security is shown in an idealized protocol model, wherein honest parties have access to a trusted party (ideal functionality). It is then shown that the environment cannot distinguish between real and ideal world simulations. The adversary is allowed to schedule the actions of honest parties, eavesdrop in communications, and interact with the environment in an arbitrary manner (Byzantine adversary).

5 Security Simulation

Initialization of honest parties. Honest parties are initialized as follows. The trusted server—symbolized by oracle $\mathcal{O}_S$—creates a database of keys $K_i, i = 1, \ldots, n$—choosing keys at random from $\{0,1\}^\tau$, where τ is a security parameter (provided as input to the initialization step). For simplicity, we do not consider dynamic corruption of tags here. Instead, the adversary is initialized with a subset of the valid keys $K_{\ell+1}, \ldots, K_n$, and so the first ℓ keys correspond to honest tags. During real-world simulations, the adversary interacts with honest tag T_i by accessing oracle $\mathcal{O}_i$, which emulates the behavior of the honest tag with corresponding key K_i.

The initialization also requires, for each ordered pair (i, j), $1 \leq i < j \leq \ell$, that one chooses two bits $b^1_{i,j}$ and $b^2_{i,j}$, independently at random. For each triple (i, j, c), with $c \in \{1, 2\}$, an *ambivalent oracle* $\mathcal{O}^c_{i\vee j}$ will use key K_i or K_j in the simulation, respectively, if $b^c_{i,j} = 0$ or $b^c_{i,j} = 1$. The role of the ambivalent oracles will soon be made clear.

As the simulation starts, each tag oracle or ambivalent oracle is marked as `available`. Each tag oracle or ambivalent oracle independently initializes values r_i, $r^c_{i\vee j}$ at random. The server $\mathcal{O}_S$ generates a random value r^0_{sys} which will be broadcast by readers as challenge to tags during the first *server period*, or simply *period*. Subsequently, the adversary may cause new periods to commence by telling $\mathcal{O}_S$ to refresh the value r^t_{sys} with a new random value, where t counts how many periods have completed before the current one.

Real simulation model. Let $\mathcal{A}$ be the adversary. $\mathcal{A}$ can internally represent many adversarial tags T' (with compromised valid keys or invalid keys) and dishonest readers R', but we represent it as a single party $\mathcal{A}$.

At the beginning of the simulation, the total number of tags n is provided to the adversary. The adversary interacts in an arbitrary manner with the simulation environment $\mathcal{Z}$. Consider a single communication session between $\mathcal{A}$ and some honest party. $\mathcal{Z}$ maintains a notion of time—we do not require synchronized clocks, $\mathcal{Z}$ only needs to discern which adversarial actions precede other adversarial actions. We now describe what types of messages can be understood by honest parties in the real protocol simulation. We note that, since individual tags execute sequentially, they are not always available to initiate new communication sessions with $\mathcal{A}$, for instance if already communicating with $\mathcal{A}$.

$\mathsf{REFRESH}()$: Called by $\mathcal{A}$ to cause the beginning of a new server period. $\mathcal{O}_S$ increments the period counter ($t \leftarrow t+1$) and generates a new random value r^t_{sys}. This value will be broadcast by honest readers as challenge to tags, until the beginning of the next server period—i.e., until another call to $\mathsf{REFRESH}()$ occurs.

$\mathsf{START}(i)$: If $\mathcal{O}_i$ is not `available`, this call is ignored. Otherwise $\mathcal{O}_i$ changes status to `communicating` with $\mathcal{A}$, and all oracles of the type $\mathcal{O}^c_{i\vee ?}$ are marked as `unavailable`.

$\mathsf{START}(i \vee j, c)$: If $\mathcal{O}^c_{i,j}$ is not `available`, this call is ignored. Otherwise $\mathcal{O}^c_{i,j}$ changes status to `communicating` with $\mathcal{A}$, and $\mathcal{O}_i$, $\mathcal{O}_j$ become `unavailable`, as well as all *other* oracles of the type $\mathcal{O}^{c'}_{i\vee ?}$, $\mathcal{O}^{c'}_{j\vee ?}$, with $c' \in \{1,2\}$.

$\mathsf{SEND}(i, m)$: If $\mathcal{O}_i$ is not `communicating` with $\mathcal{A}$ this call is ignored. Otherwise, $\mathcal{O}_i$ responds with the pair $r_i, h = H_{K_i}(m, r_i)$, and updates $r_i \leftarrow H_{K_i}(r_i)$.

$\mathsf{SEND}(i \vee j, c, m)$: If $\mathcal{O}^c_{i\vee j}$ is not `communicating` with $\mathcal{A}$ this call is ignored. Otherwise, let ι be either i or j, corresponding to whether $\mathcal{O}^c_{i\vee j}$ was initialized with key K_i or K_j, respectively. Then $\mathcal{O}^c_{i\vee j}$ responds with the pair $r^c_{i\vee j}, h = H_{K_\iota}(m, r^c_{i\vee j})$, and updates $r^c_{i\vee j} \leftarrow H_{K_\iota}(r^c_{i\vee j})$.

$\mathsf{SEND}(\mathcal{TS}, m)$: $\mathcal{O}_S$ parses m as a string $r||h$. It then consults its lookup table for an entry of the type (r, K_i). If such an entry is found, $\mathcal{O}_S$ further checks if $h = H_{K_i}(r^t_{sys}||r)$, replying to $\mathcal{A}$ with 1 (indicating authentication success) if the equality holds. If either a match is not found or the check fails, $\mathcal{O}_S$ searches its key database $\mathcal{K}$ for any K_i such that $h = H_{K_i}(r^t_{sys}||r)$. If such K_i is found, it replies to $\mathcal{A}$ with 1, or 0 otherwise. $\mathcal{O}_S$ outputs the identity i if the authentication is successful with key K_i, else it outputs nothing. This output is not observed by the environment $\mathcal{Z}$.

$\mathsf{END}(i)$: If $\mathcal{O}_i$ is not `communicating`, the call is ignored. Otherwise, $\mathcal{O}_i$ becomes `available`, as well as any $\mathcal{O}^c_{i\vee j}$ such that $\mathcal{O}_j$ is also `available`.

$\mathsf{END}(i \vee j, c)$: If $\mathcal{O}^c_{i\vee j}$ is not `communicating`, the call is ignored. Otherwise, $\mathcal{O}^c_{i\vee j}$ becomes `available`, as well as $\mathcal{O}_i$, $\mathcal{O}_j$, and any $\mathcal{O}^{c'}_{i\vee ?}$, $\mathcal{O}^{c'}_{j\vee ?}$, for $c' \in \{1,2\}$.

The role of the identity-ambivalent oracles. The ambivalent oracles $\mathcal{O}^c_{i\vee j}$ enable $\mathcal{A}$ to interact with parties whose identity is one of two possible choices. This enables attacks against anonymity, where $\mathcal{A}$'s objective is to determine if $\mathcal{O}^1_{i\vee j}$ and $\mathcal{O}^2_{i\vee j}$ represent the same or different identities. Note that the concurrency-prevention rules (enforced via the tags maintaining a status among `available`, `communicating`, and `unavailable`) are designed to prevent that $\mathcal{A}$ may disambiguate the ambivalent oracles simply on the basis of availability conflicts, while at the same time preventing that a single tag executes two sessions concurrently.

5.1 Security definitions

We now formally define the security goals of anonymous authentication protocols. We define a session *ses* with honest tag T_i as a time-interval between the first call to $\mathsf{START}(i)$ after either the beginning of the simulation or the most recent call to $\mathsf{END}(i)$, and the first subsequent call to $\mathsf{END}(i)$.

Availability holds when there is no efficient adversary $\mathcal{A}$ that during the course of the simulation, has non-negligible probability in preventing a tag T_i from authenticating itself to a reader R_j during a session ses, without changing T_i's interaction with R_j in session ses. This should remain true even if $\mathcal{A}$ has interacted with T_i or $\mathcal{TS}$ arbitrarily in the past, perhaps attempting to force either or both into an inconsistent state. Note that $\mathcal{A}$ is still allowed to interact with all other honesty parties, including reader R_j, during ses. The advantage $adv_{AVLB}^{\mathcal{A},T_i}$ of $\mathcal{A}$ in this game against T_i is the maximum probability that $\mathcal{TS}$ rejects T_i in any session.

$$adv_{AVLB}^{\mathcal{A},T_i} := Prob\,[\mathcal{TS}\,\text{rejects}\,T_i\ \text{in}\ ses \mid \mathcal{A}\ \text{only relays between}\ \mathcal{O}_i\ \text{and}\ R_j\ \text{during}\ ses]\,,$$

and $adv_{AVLB}^{\mathcal{A}}$ is defined as the maximum of the $adv_{AVLB}^{\mathcal{A},T_i}$, over all honest tags T_i in any session.

An important concern in regard to the management of RFIDs is to have a kill process, in which a reader can instruct an RFID tag to disable its functionality permanently. Current methods for disabling EPC tags have been recently shown ([36]) to allow an attacker to perform a power-analysis based recovery of the kill-key. Such attacks violate the above definition of availability. Our protocols can be adapted to support a kill-key while still guaranteeing availability.

Authentication holds when there is no efficient adversary that, during the simulation, succeeds with non-negligible probability in authenticating itself to an honest reader R_j during some session ses, and moreover: (a) The server $\mathcal{TS}$ believes $\mathcal{A}$ to have authenticated itself as tag T_i in ses; and (b) the duration interval [start-time, end-time] for session ses is disjoint from the duration intervals of all of $\mathcal{A}$'s sessions with oracle $\mathcal{O}_i$ as well as with any ambivalent oracle $\mathcal{O}_{i,j}^c$ that was initialized as $\mathcal{O}_i$. We note that in this definition, $\mathcal{A}$ is not required to know under which identity T_i it has succeeded in authenticating itself. Furthermore, it accommodates man-in-the-middle attacks, as long as the attack leads to $\mathcal{A}$'s acquiring knowledge (such as keys) that can be used for subsequent authentication attempts, while ruling out scenarios in which the adversary simply relays messages between honest parties as successful attacks. The advantage $adv_{\mathsf{AUTH}}^{\mathcal{A},T_i}$ of the adversary against authentication is simply the probability that it succeeds.

$$adv_{\mathsf{AUTH}}^{\mathcal{A},T_i} := Prob\,[\mathcal{A}\,\text{authenticates as}\,T_i\,\text{in}\,ses;\ \ ses \cap \text{Sessions}(\mathcal{A},\mathcal{O}_i) = \emptyset]\,,$$

where i is the index of an honest user. The advantage $adv_{\mathsf{AUTH}}^{\mathcal{A}}$ it the maximum of the $adv_{\mathsf{AUTH}}^{\mathcal{A},T_i}$ over all tags T_i.

Anonymity holds when no efficient adversaries have non-negligibly better-than-even chances of, at any time in the simulation, outputting a triple (i, j, b), where $1 \le i < j \le n$, and either (1) $b = 0$ and $\mathcal{O}_{i\vee j}^1 \neq \mathcal{O}_{i\vee j}^2$, or (2) $b = 1$ and $\mathcal{O}_{i\vee j}^1 = \mathcal{O}_{i\vee j}^2$. The advantage of the adversary in distinguishing T_i and T_j, $Adv_{\mathsf{ANON}}^{\mathcal{A},i\vee j}$, is defined as the difference between winning and losing probabilities when the adversarial guess bit equals 1:

$$adv_{\mathsf{ANON}}^{\mathcal{A},i\vee j} := Prob\left[(i,j,1) \leftarrow \mathcal{A} \ \mid\ \mathcal{O}_{i\vee j}^1 = \mathcal{O}_{i\vee j}^2\right] - Prob\left[(i,j,1) \leftarrow \mathcal{A} \ \mid\ \mathcal{O}_{i\vee j}^1 \neq \mathcal{O}_{i\vee j}^2\right],$$

and the adversarial advantage against anonymity, $adv_{\mathsf{ANON}}^{\mathcal{A}}$ is the maximum of the $adv_{\mathsf{ANON}}^{\mathcal{A},i\vee j}$ over all pairs (i, j), with $i < j$.

This is a unified framework because the adversary does not need to identify, at any particular point in the simulation, which security property it seeks to defeat. Instead, it may weigh its knowledge and adjust its strategy during the simulation to maximize its success in violating any of the security requirements.

Conclusion. Due to the increasing importance of ubiquitous computing applications, it is fruitful to pursue research into the attending security issues. In this overview paper, we show that it is possible to design practical, lightweight security protocols for RFIDs and MANET environments that achieve provable security within comprehensive, formal security models and that achieve availability guarantees against DoS attacks.

References

1. G. Ateniese, J. Camenisch, and B. de Medeiros. Untraceable RFID tags via insubvertible encryption. In *Proc. of the ACM Conf. on Computer and Communication Security (ACM CCS 2005)*, pp. 92–101, ACM Press, 2005.
2. B. Awerbuch, D. Holmer, C. Nita-Rotaru and H. Rubens, *An On-Demand Secure Routing Protocol Resilient to Byzantine Failures*, ACM Workshop on Wireless Security – WiSe'02 2002.
3. D. Beaver, *Foundations of secure interactive computing,* Proc. CRYPTO '91, Springer Verlag LNCS, vol. 576, pp. 377-391, 1991.
4. D. Beaver. Secure multi-party protocols and zero-knowledge proof systems tolerating a faulty minority. In *Journal of Cryptology*, vol. 4, no. 2, pp. 75122, 1991.
5. D. Beaver and S. Goldwasser. Multiparty computation with faulty majority. In *Proc. of Advances in Cryptology (CRYPTO 89)*, LNCS Vol. 435, pp. 589–590, Springer-Verlag, 1989.
6. E.M. Belding-Royer and C.-K. Toh. A review of current routing protocols for ad-hoc mobile wireless networks. In *IEEE Personal Communications Magazine*, pp. 46-55, 1991.
7. M. Burmester and T. van Le. Secure Multipath Communication in Mobile Ad hoc Networks. In *Proc. International Conference on Information Technology Coding and Computing*, pp. 405–409, 2004.
8. M. Burmester, T. van Le, and B. de Medeiros. Provably Secure Ubiquitous Systems: Universally Composable RFID Authentication Protocols. In *E-print Report #2006/131*, International Association for Cryptological Research, 2006.
9. M. Burmester, T. Van Le, and A. Yasinsac. Adaptive gossip protocols: managing security and redundancy in dense ad hoc networks. In *Journal of Ad hoc Networks*, vol. 4, no. 3, pp. 504–515, Elsevier, 2006.
10. C. Chatmon, T. Le Van, and M. Burmester. Anonymous authentication with RFID devices. *FSU Technical Report: TR-060112.* Available at *url* http://www.sait.fsu.edu/research/rfid/index.shtml.
11. R. Canetti. Studies in Secure Multiparty Computation and Applications. *Ph. D. thesis*, Weizmann Institute of Science, Rehovot 76100, Israel, June 1995.
12. R. Canetti. Security and composition of multi-party cryptographic protocols. In *Journal of Cryptology*, vol. 13, no. 1, pp. 143–202, 2000.
13. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proc. of Foundations of Comp. Sci. (FOCS 2001)*, pp. 136–145, 2001.
14. R. Canetti and M. Fischlin. Universally Composable Commitments. In *Proc. of Advances in Cryptology (CRYPTO 2001)*, LNCS 2139, pp. 19-ff., Springer-Verlag, 2001.
15. R. Canetti and J. Herzog. Universally Composable Symbolic Analysis of Cryptographic Protocols (The case of encryption-based mutual authentication and key exchange). In *E-print Technical Report # 2004/334*, International Association for Cryptological Research, 2004. Available at *url* http://eprint.iacr.org/2004/334
16. R. Canetti and H. Krawczyk. Universally Composable Notions of Key Exchange and Secure Channels (Extended Abstract). In *Proc. of Advances in Cryptology (EUROCRYPT 2002)*, LNCS 2332, pp. 337–ff., Springer-Verlag, 2002.
17. R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally Composable Two-Party and Multi-Party Secure Computation. In *Proc. of the ACM Symposim on Theory of Computing*, vol. 34, pp. 494–503, ACM Press, 2002.
18. J. R. Douceur. The Sybil attack. In *Proc. 1st International Workshop on Peer-to-Peer Systems – IPTPS '02*, 2002.

19. L.R. Ford and D.R. Fulkerson. *Flows in Networks.* Princeton University Press, *Princeton*, NJ, 1962.
20. H. Gilbert, M. Rodshaw, and H. Sibert. An Active Attack Against HB+ – A Provably Secure Lightweight Authentication Protocol. *PerSec '04*, March 2004. Full paper available in *E-print Technical Report # 2005/237*, International Association for Cryptological Research, Available at *url* http://eprint.iacr.org/2005/237.pdf
21. M. Hirt and U. Maurer. Player Simulation and General Adversary Structures in Perfect Multiparty Computation. In *Journal of Cryptology*, Vol. 13, No. 1, pp. 31–60, 2000.
22. D. Hofheinz, J. Müller-Quade, and R. Steinwandt. Initiator-Resilient Universally Composable Key Exchange. In *Proc. of the European Symp. on Research in Computer Security (ESORICS 2003)*, LNCS 2808, pp. 61–84, Springer-Verlag, 2003.
23. N. J. Hopper and M. Blum. Secure Human Identification Protocols. In *Proc. of Advances in Crypotology (ASIACRYPT 2001)*, LNCS, Springer-Verlag, 2001.
24. Y-C Hu, D.B. Johnson and A. Perrig. Ariadne: A Secure On-Demand Routing protocol for Ad Hoc Networks. In *Proc. of the ACM Annual Intern. Conf. on Mobile Computing and Networking (MobiCom 2002)*, ACM Press, 2002.
25. Y-C Hu, D.B. Johnson and A. Perrig. SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks. In *Proc. 4th IEEE Workshop on Mobile Computing Systems & Applications (WMCSA 2002)*, IEEE, Calicoon, NY, 2002.
26. Y-C. Hu, A. Perrig and D.B. Johnson. Rushing attacks and defense in wireless ad hoc network routing protocols. In *Proc. of WiSe2003*, pp. 30–40, 2003.
27. D.B. Johnson and D.A. Maltz. Dynamic Source Routing in Ad-Hoc Wireless Networks. In *ed. T. Imielinski and H. Korth, Mobile Computing*, Kluwer Academic Publisher, pp. 152–181, 1996.
28. A. Juels and S. A. Weiss. Authenticating Pervasive Devices with Human Protocols. In *Proc. of Advances in Cryptology—CRYPTO 2005*, LNCS vol. 3621, pp. 293–ff, Springer-Verlag, 2005.
29. J. Katz and J. S. Shin. Parallel and Concurrent Security of the HB and HB+ Protocols. To appear in *Proc. of Advances in Cryptology (EUROCRYPT 2006)*, Springer, 2006.
30. P. Laud. Formal analysis of crypto protocols: Secrecy types for a simulatable cryptographic library. In *Proc. of the 12th ACM Conf. on Computer and Communications Security (ACM CCS 2005)*, pp. 26–35, ACM Press, 2005.
31. P. Papadimitratos and Z.H. Haas. Secure Routing for Mobile Ad hoc Networks. In *Mobile Computing and Communications Review*, Vol 6, No 4, 2002.
32. C. E. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing for Mobile Computers. In *Computer Communications Review*, pp. 224-244, 1994.
33. C.E. Perkins and E.M. Royer. Ad hoc on-demand distance vector routing. In *Proc. of the IEEE Workshop on Mobile Computing Systems and Applications*, pp. 90–100, 1999.
34. B. Pfitzmann and M. Waidner. Composition and integrity preservation of secure reactive systems. In *Proc. of the ACM Conf. on Computer and Communications Security (ACM CCS 2000*, pp. 245–254, ACM Press, 2000.
35. B. Pfitzmann and M. Waidner. A model for asynchronous reactive systems and its application to secure message transmission. In *Proc. of the IEEE Security and Privacy Symposium (S & P 2001)*, pp. 184–200, 2001.
36. Y. Oren and A. Shamir. Power Analysis of RFID Tags. *Invited talk, RSA Conference, Cryptographer's Track (RSA-CT 2006).* Available at `http://www.wisdom.weizmann.ac.il/~yossio/rfid`
37. A.J. Menezes, P.C. van Oorschot and S.A. Vanscott. *Handbook of Applied Cryptography*, CRC Press, 1996.
38. G. Tsudik. YA-TRAP: Yet another trivial rfid authentication protocol. *International Conference on Pervasive Computing and Communications*, 2006.
39. M. G. Zapata. Secure Ad hoc On-Demand Vector (SAODV) Routing. *IETF Internet Draft.* Available at *url* http://www.potaroo.net/ietf/all-ids/draft-guerrero-manet-saodv-00.txt (Work in Progress).